Handling Errors

When an exception occurs in your handler function, the RunPod SDK automatically captures it, marking the job status as `FAILED` and returning the exception details in the job results.

Implementing custom error responses

In certain scenarios, you might want to explicitly fail a job and provide a custom error message. For instance, if a job requires a specific input key, such as *seed*, you should validate this input and return a customized error message if the key is missing. Here's how you can implement this:

```
import runpod


def handler(job):
    job_input = job["input"]

    # Validate the presence of the 'seed' key in the input
    if not job_input.get("seed", False):
        return {
            "error": "Input is missing the 'seed' key. Please include a seed and retry your request."
        }

    # Proceed if the input is valid
    return "Input validation successful."


# Start the RunPod serverless function
runpod.serverless.start({"handler": handler})
```

> **NOTE**
>
> Be cautious with `try/except` blocks in your handler function. Avoid suppressing errors unintentionally. You should either return the error for a graceful failure or raise it to flag the job as `FAILED`.

One design pattern to consider, is to Refresh your Worker when an error occurs.

Edit this page